NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INTELLIGENT TUTORING SYSTEMS:
A DESIGN SUPPORT TOOL

by

Harry Edward Landau

September, 1994

Thesis Advisor: Kishore Sengupta

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1994	Master's Thesis

4. TITLE AND SUBTITLE
INTELLIGENT TUTORING SYSTEMS: A DESIGN SUPPORT TOOL

5. FUNDING NUMBERS

6. AUTHOR(S)
Landau, Harry E.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE
A

13. ABSTRACT *(maximum 200 words)*

The Department of Defense (DoD) is facing a shortage of qualified instructors for its advanced technical weapons training facilities. This shortage is because of the current downsizing trend, without a similar trend in the development of weapons and computer systems. To maintain a force of highly trained, technical personnel, DoD must investigate other methods for training and maintaining a highly technical force. The advanced capabilities of modern computer systems with the use of Intelligent Tutoring Systems (ITS) can provide a supplemental training aid for the lack of human instructors. This thesis proposes the use of a Design Support Tool (DST) to assist instructional designers in the development of ITS systems for the DoD.

14. SUBJECT TERMS
Intelligent Tutoring System, Computer Aided Instruction, Design Support Tools

15. NUMBER OF PAGES
54

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

INTELLIGENT TUTORING SYSTEMS:

A DESIGN SUPPORT TOOL

Harry Edward Landau

Lieutenant Commander, United States Navy

B.A., California State University, Northridge, 1976

Submitted in partial fulfillment of the

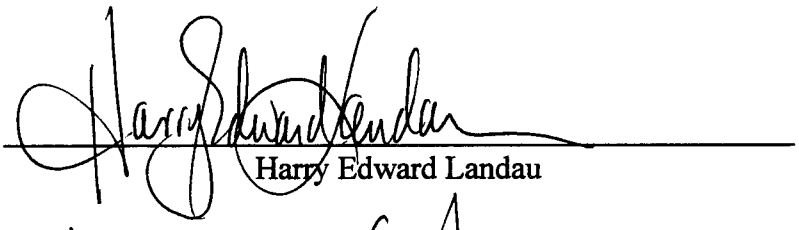
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

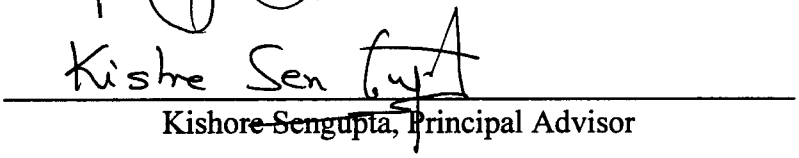
from the

NAVAL POSTGRADUATE SCHOOL

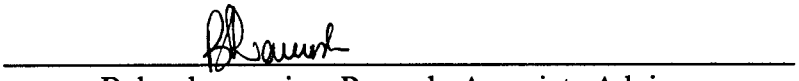
September 1994

Author: 

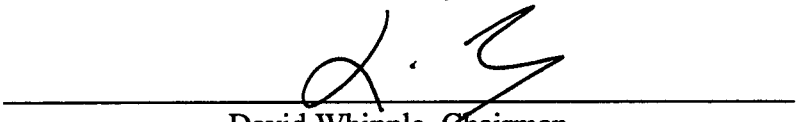
Harry Edward Landau

Approved by: 

Kishore Sengupta, Principal Advisor



Balasubramaniam Ramesh, Associate Advisor



David Whipple, Chairman

Department of Systems Management

ABSTRACT

The Department of Defense (DoD) is facing a shortage of qualified instructors for its advanced technical weapons training facilities. This shortage is because of the current downsizing trend, without a similar trend in the development of weapons and computer systems. To maintain a force of highly trained, technical personnel, DoD must investigate other methods for training and maintaining a highly technical force. The advanced capabilities of modern computer systems with the use of Intelligent Tutoring Systems (ITS) can provide a supplemental training aid for the lack of human instructors. This thesis proposes the use of a Design Support Tool (DST) to assist instructional designers in the development of ITS systems for the DoD.

DISCLAIMER

The views expressed in this thesis are those of the author and do reflect the official policy or position of the Department of Defense or the U.S. Government.

TABLE OF CONTENTS

I. INTRODUCTION

Due to the current downsizing of the Department of Defense (DoD) and continued increase in the need for highly trained technical personnel, DoD will face a shortage of qualified instructors to train and maintain this technical force. The DoD must find an means to supplement the lost of these instructors for training its personnel.

Possible solutions would be to hire civilian instructors or add provisions in system maintenance contracts for continuous training from the contractor. Neither solution eliminates the basic problem: not enough funds to supply the required instructors. Another alternative would be to supplement current and future training with Computer Aided Instruction (CAI).

CAI can provide training without the additional expense of an instructor at every site. CAI, developed with modern authoring systems, can also take advantage of current computer multimedia capabilities providing students with better training aids than the textual CAI of the past. These authoring systems can provide excellent training, with one exception -- no human instructor is available to interact with the student.

The human instructor can sense when a student does not understand the material being presented and can ask probing questions to determine the nature of the student's difficulty. Most current CAI lacks this ability and simply presents the material in a predetermined path, providing minimal feedback to the student. The student may, if desired, repeat a lesson. However, this repetition may not provide the additional explanations or information the student requires to understand the training material. A

human instructor can rephrase the information or present different information to support the training objectives.

S. Ohlsson states:

> ... The task of tutoring is forever changing. School systems undergo reforms, student populations follow the rhythms of the surrounding society, topics are added to (or deleted from) the curriculum, courses are moved from one age-level to another, and course contents vary from textbook to textbook, from classroom to classroom. What we need then are not particular, quickly outdated, computer tutors but the know-how of tutor construction. ... Even a pedagogically successful tutor has limited interest if it leaves system designers without any information about how to design the next system. *The output of research into computer tutoring should consist, not of particular systems, but of principles which allow specifications - in terms of course content and student characteristics - to be turned into effective tutors* (emphasis added) [Ref. 5].

What is needed is a system that supplements the human instructor, is able to determine the student's level of comprehension, and tailors the instruction to the student's ability.

A. INTELLIGENT TUTORING SYSTEMS

Intelligent Tutoring Systems (ITS) have evolved from the earlier Computer Aided Instruction (CAI) systems. The early CAI systems were basically page-turners, presenting prepared text, or drill-and-practice programs. These systems presented problems and responded to the students' solutions using stored answers and remedial comments [Ref. 8]. To improve on the value of using computers in education, software developers introduced artificial intelligence techniques in the traditional CAI systems. These systems were referred to as Intelligent Computer Aided Instruction (ICAI) system, which later became known as Intelligent Tutoring Systems (ITS) [Ref. 10].

Burns and Capps [Ref. 3] state that a CAI must pass three tests before it can be considered an Intelligent Tutoring System:

1. The computer must understand the subject matter well enough to draw inferences or solve problems in the subject matter or domain.

2. The system must be able to deduce a learner's level of comprehension of the knowledge contained within the subject matter.

3. The strategy of the ITS must be intelligent; that is, the "instructor in a box" can implement strategies to reduce the difference between expert and student performance.

An ITS utilizes an information network of facts, concepts, and procedures. It should automatically generate questions, explanations, and answers [Ref. 10]. The ITS resembles and attempts to replicate, a one-on-one session between a student and instructor. During a session, the ITS will attempt to determine the current state of the student's level of comprehension and alter the tutoring session to accommodate this model.

VanLehn [Ref. 11] describes several essential problems of student modeling in ITSs. He refers to these problems as the bandwidth of available knowledge (how much of the student's activity is accessible to the system for diagnostic purposes) the types of knowledge to be learned, and the ability of the system to assess the differences between the student and the expert.

VanLehn classifies knowledge into two types: procedural, which is sub-typed into flat and hierarchical, and declarative. Procedural knowledge is used to perform tasks, while declarative knowledge is fact-like and not specialized. Flat knowledge presents information based on preset rules and the current actions by the student. Hierarchical knowledge allows for "subgoaling" [Ref. 11]; however, this knowledge system is harder to code and needs to infer conditions, problem states and subgoals [Ref. 11].

This thesis recommends the use of a student information module to map the student's usage of the ITS instead of a diagnostic system designed to generate a model of a student. This module is discussed in Section II.D.

ITSs can be designed to provide different aspects of a training environment. Some ITSs replace the instructor and provide an environment that can be used without any human instructor/tutor assistance. Other ITSs supplement, rather than replace, the human instructor. The goals of such an ITS are to:

1. Assist the instructor to organize lessons for a course.

2. Provide consistency among the lessons, providing the instructor with a system that allows for the ITS to be updated with new material and maintain relevance to material being presented in the classroom.

3. Provide students with an environment that supplements the classroom instruction. The goal is not to replace the classroom instructor, but allow to students another means to review and obtain additional information presented in the classroom.

The basic ITS proposed by this thesis (Figure 1-1) is modeled after the second type of ITS, which assists the instructor in designing and maintaining a course of instruction and supplements the classroom instruction. The main component is the interface, or system moderator, which provides the interface between all of the other

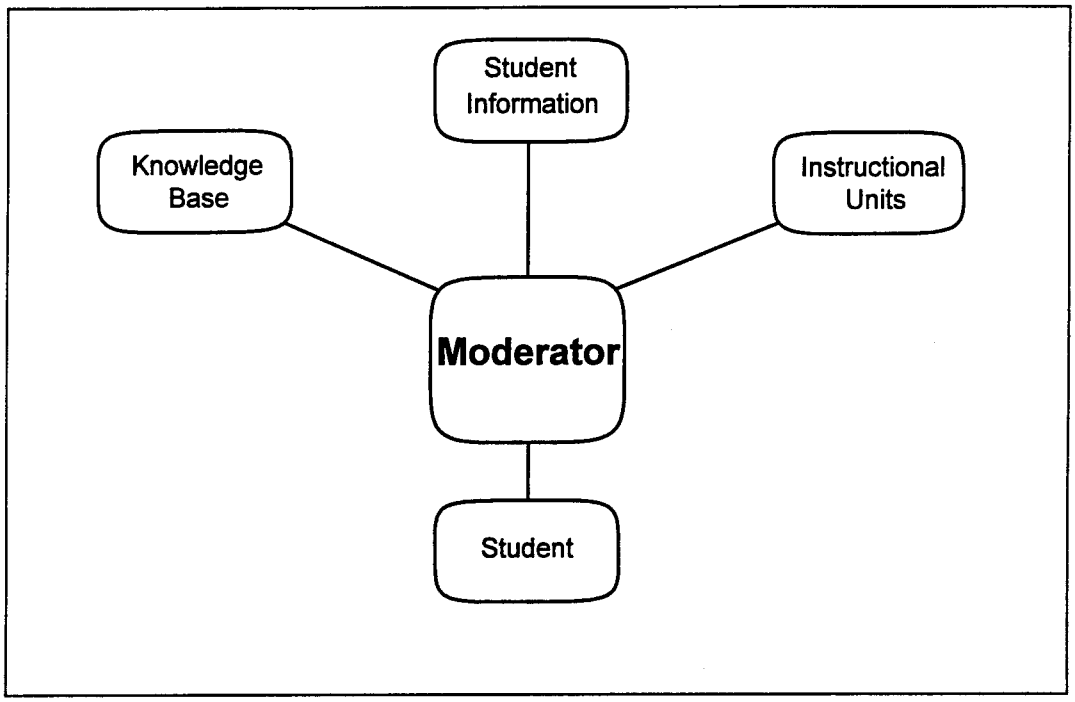


Figure 1-1: Basic Intelligent Tutoring System Model

components of the ITS. The moderator communicates directly with the student using the system and transmits messages between the other three components. The Instructional Unit (IU) module contains the actual instruction to be presented to the student. The Knowledge Base module is an expert system consisting of set of rules by which the ITS determines what, if any, additional IUs the student requires. The Student Information module is generated as the student uses the system and is used to aid the moderator in

determining what IUs to execute. This module also provides information to the instructor about the student's progress.

B. THEORETICAL PREMISE

Most current research aims to have the ITS replace authoring systems and do all of the tasks necessary to provide all training on the subject being presented (MYCIN, ACT*, etc.) [Refs. 1, 4, 12]. This approach limits the ITS to a specific subject. A goal of ITS research should be the development of tools to aid the instructor in designing a course of instruction. The proposed Design Support Tool (DST) would allow the instructor to design individual components of the ITS, using the best tools (*i.e.*, authoring systems, expert systems, etc.) for each section of the ITS, develop an easy interface to link these components together, and maintain the ITS up to date.

C. METHODOLOGY

The design support tool should provide an easy interface for a designer to integrate the components of an ITS into a complete tutorial system. The components of an ITS are the individual lessons generated by an authoring tool, the knowledge base generated by or included in an expert system, the student module generated by the tutoring system as a student uses the system, and a system moderator which interfaces with all the components and directly interacts with the student using the system. Instead of incorporating all of these components into one package, this thesis proposes a system composed of a design tool and system moderator which will integrate the end product of other tools. This paper will show what should be included in the design tool and how the

tool should be used to aid in the integration the individual components to provide the best tutorial system.

II. DESIGN COMPONENTS

This chapter describes the four components of a basic ITS and how they interact with each other and the user.

A. MODERATOR

The moderator is the heart of the ITS. Its function is to interface directly with the student, execute the selected IUs, pass and receive messages between the individual modules (IUs, Student, Knowledge Base) and maintain a log of the student's usage and level of comprehension. An instructional designer can use the proposed Design Support Tool (DST) to develop the structure of the instructional material. The moderator can use this along with the information gained during the actual use of the ITS to determine the flow of the tutorial.

B. INSTRUCTIONAL UNITS

The basic building block of a tutorial is the Instructional Unit (IU). IUs are the actual components or lessons developed using an authoring system and whose main function is to present the instructional information to the students [Ref. 6]. This information may be presented by the IU through a combination of different formats; *i.e.*, textual, audio, graphical, video, etc.

An example of a simple IU would be an introduction to a topic. This type of IU may be text only and simply explains how to use the system and states the learning objectives for the topic. The interaction with the student may also only be page turning and not incorporate multimedia techniques.

A more complex IU would incorporate more than just text. For example an IU may use a multimedia presentation to teach the components of the control console of an airplane. This type of IU would use audio, video and computer-generated graphics. The audio would explain what the student is viewing. The video would be a view looking out from the cockpit, and the computer graphics would simulate the control console instruments. The IU would demonstrate the appearance of the instruments when climbing, diving, or turning or would instruct the student in the techniques needed for take-off and landing.

The IUs function by communicating simultaneously with the student and the moderator. The student in turn also simultaneously communicates with the IUs and the moderator. Student performance dictates which IU is appropriate. The IUs must have the capability to communicate directly with the student, periodically send messages to the moderator, receive instructions, and perform required actions in order to provide the instructional material to the student. For example when the student requests a specific lesson to be executed, the student interacts directly with the moderator which verifies that the request can be accomplished. The moderator loads and executes the IU which interacts directly with the student, presenting the training and receiving input (*i.e.* mouse button actions, menu selections, etc.) and transfers this information to the moderator in the form of a message. Other messages are generated when the IU starts execution and again when it completes its execution.

C. KNOWLEDGE BASE MODULE

The Knowledge Base Module of the ITS is a set of rules designed to simulate the presence of a human instructor or tutor. These rules are compiled with an expert system generator and assist the moderator in determining the student's level of comprehension.

The knowledge base rules may be simple control rules or more complex multilevel rules. A simple control rule may verify that the student has already used a specific course of instruction and therefore skip an introductory IU:

```
if FirstTime
 then execute intro.exe
```

A complex rule might determine if a student's request to execute a specific IU is valid:

```
execute lesson2 if
        lesson1 complete
and
        lesson1quiz complete with result > .85
```

The system moderator receives messages from the IUs and sends them to the knowledge base module. The knowledge base module interprets the messages using a set of rules designed for the specific course of instruction and returns a message to the moderator. The moderator uses the message from the knowledge base module to build a picture of the student's level of comprehension and makes decisions as to which IUs to execute during the tutoring session.

D. STUDENT INFORMATION MODULE

The Student Information Module, maintained by the moderator, contains a set of information which the system infers as the student uses the system. The information

contained in this module helps the system determine the student's level of comprehension. It includes a log of the student's usage of the system in order for system monitors (instructors or remote site training officers) to assist the student's training.

This Module would contain the following student information:

1. Student name
2. Student Social Security Number
3. Lesson Plans assigned to the student
4. For each Lesson Plan assigned:
    a. Date and time plan started
    b. List of lessons completed with date and time started and completed
    c. Results of quizzes or exams taken by student
    d. Information in a free format from the student (*i.e.*, questions about training material, problems encountered with system, errors noted in training material, etc.).

This information module is maintained in real time based on messages received by the system moderator from the IUs, the knowledge base module, and from the student. The moderator will not try to build a model of the student, but maintains information about the student's usage of the system.

III. DESIGNING AN INSTRUCTIONAL DESIGN SUPPORT TOOL

A. DESIGN SUPPORT TOOL

Russell *et al.* [Ref. 6] describe an Instructional Design Environment (IDE), or Design Support Tool (DST), as "an interactive design and development system to help instructional designers deal with the complexity of creating instructional materials." The objective of this tool is to provide an instructional course designer with the ability to design and generate a complete ITS, and allow for future modifications as the course changes or as deemed necessary to configure specific courses to individual students.

IDE does not provide links between different components generated by other software packages. Incorporating this functionality should help the instructor to formulate lesson plans and courses by allowing visual examination of the interrelationships being developed in the course design. It should also provide a repository of information and course components (*i.e.*, IUs) that is available to an instructor or instructional designer for designing different versions of topics. These versions would allow the instructional designer to provide different levels of training to different students and use the same IUs for different courses.

This method of designing an ITS allows the designer to utilize the best features of an authoring tool to develop the actual IUs and an expert system generator to develop the knowledge base component. Once these components are developed, the ITS generated by the proposed DST will use the computer's multitasking features to execute the ITS.

The DST should provide a user interface that allows the designer to generate an ITS similar to the way a course of instruction is developed for a classroom environment. The DST design presented in this thesis breaks down the design process into logical steps. The designer may use the DST without having completed the development of IUs or the knowledge base component and may edit or modify the ITS as the individual components are developed.

B. PROPOSED DESIGN SUPPORT TOOL

The proposed DST is divided into separate object-oriented classes: Lessons, Lesson Plans, and Students. These classes have associated class browsers which provide the interface with the instructional designer when developing or designing a course of instruction for an ITS. The following sections in this chapter will describe how an instructional designer would use the DST to develop an ITS. A prototype of the DST was devloped using SmallTalk from ParcPlace on a Sun Workstation. Chapter IV explains the process for developing a course of instruction using an example.

The Design Support Tool will allow the designer to define a group or groups of lessons with associated IUs and assign them to individual lesson plans that are then assigned to individual students. The resulting flexibility in the development of an ITS allows the designer to customize a set of courses for individuals and to maintain the instruction up-to-date simply by modifying an IU and, if necessary, the lesson plan for a specific course.

1. Defining Lessons

The lesson, the basic class of the DST, is the link between the moderator and the IUs. Lessons are not the same as an IU. The IU is the actual material, or executable file, that presents the instructional information to the student and is generated by an authoring system. Lessons are objects which contain information about a specific IU and allow the designer to determine which IUs to link into a lesson plan.

The instructional designer must determine how many IUs are required to present the information necessary to meet the learning objectives. The number of IUs will depend on the number of course sections. The designer may also include additional IUs to expand the basic IUs. The designer generates a library of IUs and then uses the DST's Lesson Browser (Figure 3-1) to link these into a complete tutorial.

The Lesson Browser screen is divided into two sections. The upper section (Window 1, Figure 3-1) displays all of the lesson objects that have been generated, while the lower section (Window 2, Figure 3-1) displays information about a selected lesson object.

The lesson class as defined by the DST consists of information necessary for the instructional designer to determine where, how and if he will use a specific IU in the ITS that he is designing. This information (Window 2, Figure 3-1) consists of: (a) a name for the lesson, (b) the type of lesson, (c) a short description, (d) details pertinent to the lesson and (e) the executable file name for the IU.

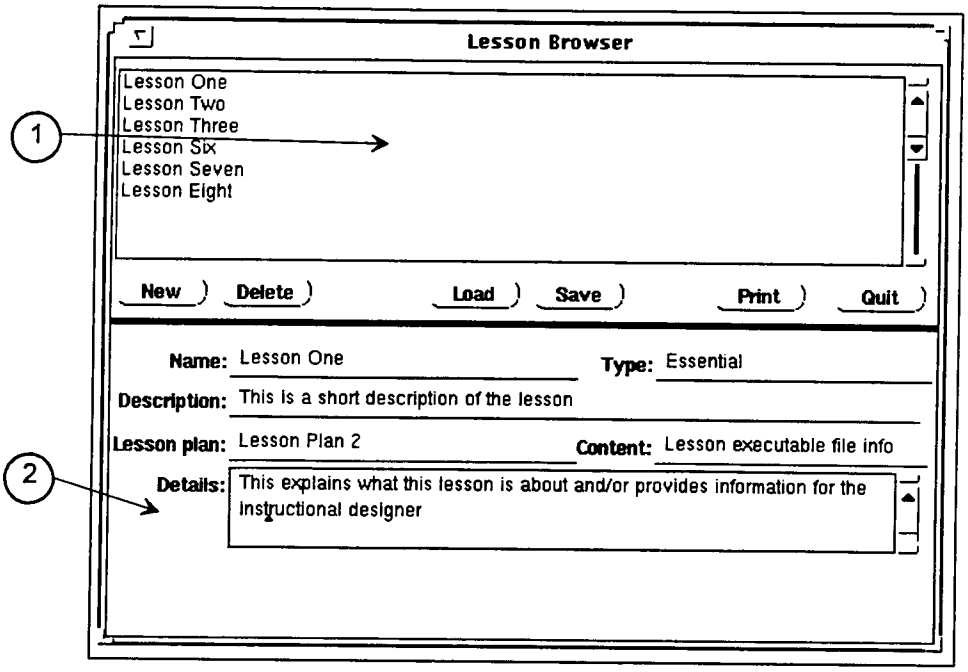


Figure 3-1: Lesson Class Browser

The name of the lesson should be unique. The DST prompts the designer if an attempt is made to add a new lesson with the same name as an existing lesson. (The lesson name is not case sensitive.) A short description provides the designer with basic information about the lesson, and a detailed description provides an area to explain how the lesson is used, any special features of the lesson, etc. The type of lesson specifies the importance of the lesson. Examples of lesson types are: Essential, Important, or Desirable. Essential lessons must be included in the instruction, and the moderator of the ITS will always present these IUs to the student. Important lessons may be skipped if the ITS determines that the student has mastered the requisite knowledge. Desirable lessons

are presented to a student if that student shows a major lack of knowledge or specifically requests additional information.

2. Defining Lesson Plans and Assigning Lessons

Once the lessons are designed, the designer must determine how to combine them to develop the course outline. The Lesson Plan Class provides the means to link lessons together. A course of instruction may contain one or several Lesson Plans. The designer reviews the lessons and formulates a plan for the course, linking individual lessons together with the Lesson Plan Browser (Figure 3-2). The Lesson Plan Browser provides the designer with all of the information needed to design the plan and link the individual lessons together. Window 1 is a list of all Lesson Plans generated. These plans may or may not be complete. The window provides a pop-up menu to add or delete, compile, print, save, load or graph a lesson plan (Refer to Section B.3) When a plan is selected, windows 2, 3, and 5 are updated, and any available data are displayed.

Window 2 provides a description of the selected plan. This window also allows editing any of the information about the plan.

Window 3 lists all the lessons assigned to the plan selected in window 1. This window provides the user with a pop-up menu that allows adding or deleting lessons. The DST will not allow the user to add the same lesson more than once and will display an error screen to inform the user. When a lesson is selected, windows 4 and 6 update and display any information available about the selected lesson.

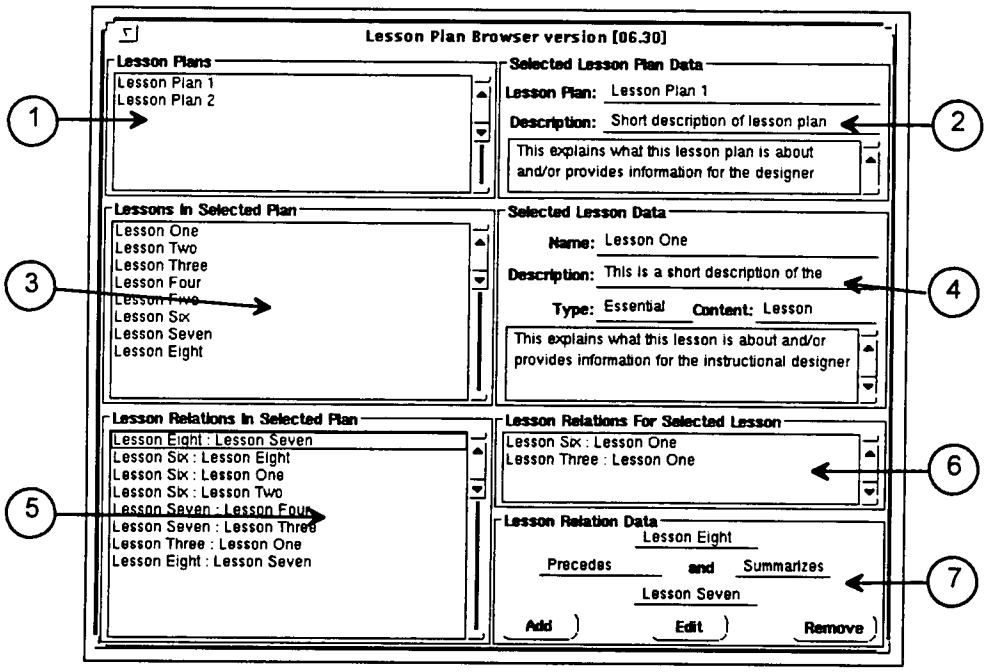


Figure 3-2: Lesson Plan Class Browser

Window 4 is a read only window providing detailed information about a selected lesson in window 3.

Window 5 is a read-only window, listing all assigned relationships for the selected plan (refer to Section B.3 for more information about relationships). When a relationship is selected, it is displayed in window 7.

Window 6 is a read-only window, displaying all of the assigned relationships for the selected lesson. When a relationship is selected, window 7 displays the information about the relationship.

Window 7 displays the selected relationship between two lessons. The information displayed may be an entry selected in window 5 or 6. The information displayed will always be from the last selection made. If the last selection is de-selected

and a selection remains in the other window, window 7 will update to display information about that selection. This window is used to add, delete or edit lesson relationships, discussed in the next section.

3. Assigning Relationships Between Lessons

Once all of the lessons have been assigned to a Lesson Plan, the designer must establish the relationships between the individual lessons. These relationships are used to determine the order of precedence of the lessons and how each lesson relates to other lessons. Using the Lesson Plan Browser, the designer selects two lessons to form a relationship between (i.e., Lesson Eight and Lesson Seven in Window 7, Figure 3-2). The designer determines which lesson precedes the other and assigns a precedence of "PRECEDES" or "SUCCEEDS". Once the order of precedence is assigned, the designer assigns a relation, between the two lessons, such as EXPLAINS, EXPANDS, SUMMARIZES, DEMONSTRATES, NONE, etc. Relationships between lessons must have both a precedence and a relation. The relation must inform the DST the order in which the lessons are to be placed and the relationship between the lessons. This two-dimensional aspect of relationships allows the DST to make consistency checks and ensure that the plan will function logically. Table 3-1 shows what type of inconsistencies are not allowed by the DST. The DST will attempt to solve any inconsistentcy or conflict in new or revised relationships at the time that they are added or changed.

Once the all of the relationships between all of the lessons have been assigned, the designer then uses the DST to generate a Lesson Plan Tree Relation graph (Figure

	Relations	Problem	Correction
Existing relation: Add new relation:	L1 Precedes & Summarizes L2 L1 Precedes & Summarizes L2	New relation same as existing relation	System displays error message and disregards new relation
Add new relation:	L1 Precedes & Summarizes L1	Both lessons of new relation are the same	System displays error message and disregards new relation
Existing relation: Add new relation or edit existing relation to:	L1 Precedes & Summarizes L2 L2 Precedes & Summarizes L1	Conflict in the existing relation and the new one. L1 cannot precede L2 and be preceded by L2 jointly.	System displays both old and new relations. User must select one relation to be retained by the system.
Existing relation: Add new relation or edit existing relation to:	L1 Precedes & Summarizes L2 L1 Precedes & Explains L2 or L1 Succeeds & Summarizes L2	Two relations with the same lessons and different precedence and/or relation.	System displays both old and new relations. User must select one relation to be retained by the system.
Existing relations: Delete lesson L2 from Lesson Plan:	L1 Precedes & Summarizes L2 L2 Precedes & Summarizes L3	Removing L2 breaks the connections between L1 & L2 and L2 & L3. There is no connection between L1 & L3.	System displays error message. User must explicitly make a new relation to connect L3 to another lesson. If no connection is made, system will display error message when compiling Lesson Plan.

NOTES:
1. L1, L2 and L3 represent three different lessons in the selected Lesson Plan.
2. An existing relation is one that has been added to the Lesson Plan prior to the user request (i.e.; Add new relation:).

Table 3-1: Relation Consistency Checks

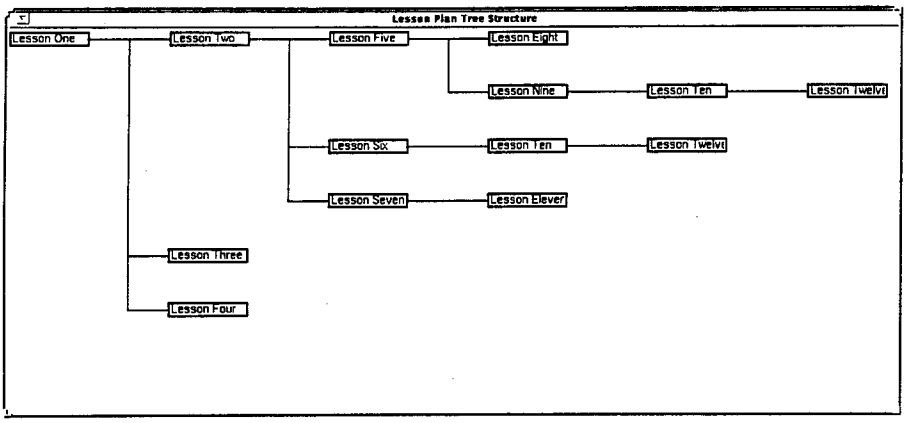


Figure 3-3: Lesson Relation Graph

3-3), which allows the designer to view the overall structure of the Lesson Plan and make any changes deemed necessary to improve the flow of the instruction. When the DST is compiling the tree it captures any remaining inconsistencies within the plan. These remaining inconsistencies occur when a lesson does not relate to any other lessons in the plan, a lesson linked to more than one lesson in an endless loop, etc.

4. Updating or Modifying Lessons and Lesson Plans

An advantage of using the DST is that the designer can easily view the completed design of the tutorial and, if necessary, change the precedence or relationship of the lessons to meet any design changes or to configure a Lesson Plan for a specific student. The DST allows for the easy alteration of the entire ITS without requiring the original tools used to generate the instructional units. If the content of the IU needs to be

modified, the authoring system would be required. Only selected IUs are updated; the rest of the system remains intact.

5. Assigning Lesson Plans to Students

Once individual Lesson Plans are developed, an individual instructor or training officer assigns Lesson Plans to the students. To assign a Lesson Plan, the person responsible for assigning training uses a Student Class Browser (Figure 3-4). The Student Class Browser shows which students are already logged into the DST system (Window 1, Figure 3-4) and if necessary adds or deletes a student or students. When a student is selected in this window, a list of assigned lesson plans is displayed in Window 2.

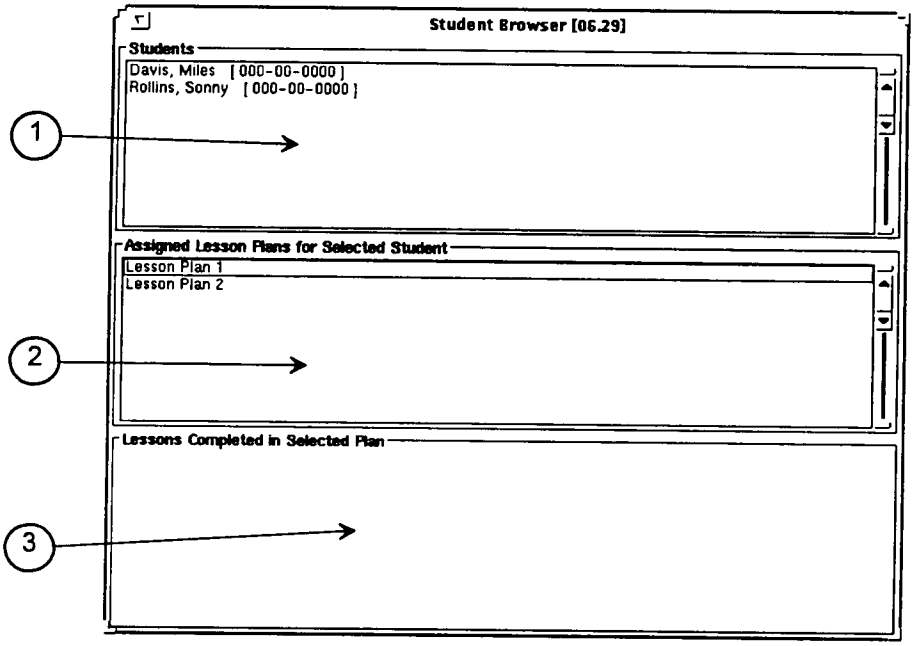


Figure 3-4: Student Class Browser

Window 2 of the browser allows the training individual to assign or remove individual lesson plans for the selected student. The DST only allows a student to be assigned one version of a lesson plan and displays an error message if an existing lesson plan is requested to be added a second time. When a lesson plan is selected in this window, the DST will display any lessons flagged as completed in Window 3. Window 3 is a read-only window and only provides information to the user of the browser.

Once the lesson plans are assigned to individual students, the student has access to that plan. When the student is logged onto the ITS moderator, a screen showing what lesson plans are available for execution is displayed.

IV. EXAMPLE

A. SAMPLE COURSE

This chapter illustrates how to design a sample course of instruction using the proposed DST. The sample course is based on Timothy Budd's book AN INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING [Ref. 2].

Budd prefaces the book by stating that he wrote it was because he "was faced with teaching a course and was not able to find a suitable existing text" that met his objectives for the course. His objectives for the course were to teach theconcepts of object-oriented programming (OOP) and provide examples of their implementation in different languages. He therefore set out to design both a text and course to teach OOP.

The book is divided into twenty chapters, each with several sections. Using the framework previously outlined, the book can be viewed as a course that contains various lesson plans (chapters), each of which contains various lessons (sections).

Each chapter (lesson plan) has a short introduction that presents the objectives for that chapter. Each section (lesson) provides the student with the knowledge necessary to meet the objectives. The contents of the lessons can vary widely: Some of the lessons are simple text explanations, while others provide graphical displays and examples of code in different languages.

This illustration uses Chapter 6, "Classes and Methods", for its material.

B. DESIGNING THE LESSONS

Chapter 6 is divided into six sections, some with subsections. The structure of the chapter is as follows:

6.0 Inheritance
- 6.1 The Benefits of Inheritance
  - [6.1.1] Software Reusability
  - [6.1.2] Code Sharing
  - [6.1.3] Consistency of Inheritance
  - [6.1.4] Software Components
  - [6.1.5] Rapid Prototyping
  - [6.1.6] Polymorphism
  - [6.1.7] Information Hiding
- 6.2 The Cost of Inheritance
  - [6.2.1] Execution Speed
  - [6.2.2] Program Size
  - [6.2.3] Message-Passing Overhead
  - [6.2.4] Program Complexity
- 6.3 Heuristics for When to Subclass
  - [6.3.1] Specialization
  - [6.3.2] Specification
  - [6.3.3] Construction
  - [6.3.4] Generalization
  - [6.3.5] Extension
  - [6.3.6] Limitation
  - [6.3.7] Variance
  - [6.3.8] Combination
- 6.4 If It Ain't Broke
- 6.5 Tree Versus Forest
- 6.6 Composition Versus Construction

The overall learning objective of chapter 6 is to understand inheritance and the use of inheritance in objected-oriented programming. The chapter can be considered a single lesson plan dealing with the OOP topic of inheritance. When defining a lesson plan the designer should include all of the IUs which support the learning objectives of

that topic. The subsections of chapter 6 are individual lessons each intended to present material which provides information pertinent to the learning objectives. Some lessons are designed to expand on the information presented in a previous lesson either by explaining in more detail or presenting an example. Other lessons are designed to summarize a topic or sub-topic while others are used to test the student's comprehension of the material presented. Each of these different types of lessons are linked together to form the overall lesson plan. Each link between lessons requires a precedence and relation for the DST to compile the lesson plan into a workable structure for the ITS moderator.

Chapter 6 can be designed as an individual lesson plan to present the IUs necessary for the topic of OOP inheritance. Each of the chapter's subsections are individual lessons and present information necessary to fulfill the learning objectives.

The instructional designer must determine which of the lessons are essential to the learning objectives and so indicate when using the DST. These lessons are then required to be presented, no matter what the ITS determines is the student's level of comprehension. Using the example, the essential lessons are each of the main subsections (*i.e.*: The Benefits of Inheritance, The Cost of Inheritance, etc.) which will be presented to all students using the ITS.

The next set of lessons important to the learning objectives, but which may be skipped if the student meets certain criteria, must be identified. These lessons are designated by the designer and the criteria placed in the knowledge base rules. The

determination of the criteria is based on what each lesson will present and how much it expands on a previous lesson. Table 4-1 shows some sample relations for this lesson plan.

C. USING THE DESIGN SUPPORT TOOL

Once most of the design questions have been answered, the instructional designer can either use the DST first or wait until after developing the individual IUs. Using the DST first will allow the designer to experiment with different lesson plan structures before committing a final design to the IUs. When using the DST, the designer first accesses the Lesson Browser and enters the information for each lesson. To do this, the designer executes the browser, select the "New" push button, and enter the information about the lesson. The lesson for section 6.0 is entered as follows:

1. Start the Lesson Browser.
2. Press "New" push button.
3. Enter a name for this lesson: "Introduction to Inheritance."
4. Enter the type of lesson: "Essential."
5. Enter a short description: "Introduction to inheritance in OOP."
6. Ignore Lesson Plan (This is used by the DST).
7. Enter the file name of the IU: "L6-0.EXE."
8. Enter a detailed description of this lesson.

This procedure is done for each lesson; a sample Lesson Browser screen would look like Figure 4-1.

Lesson 1	Precedence	Relation	Lesson 2	Reason
The Benefits of Inheritance	Succeeds	Expands	Inheritance	"The Benefits of Inheritance" expands on the reason for inheritance by explaining the benfits of using inheritance.
Software Reusability	Succeeds	Explains	The Benefits of Inheritance	"Software Reusability" explains one of the benefits of using inheritance in OOP.
Code Sharing	Succeeds	Explains	The Benefits of Inheritance	"Code Sharing" explains one of the benefits of using inheritance in OOP.
If It Ain't Broke	Succeeds	Expands	Inheritance	"If It Ain't Broke" expands on the reason for inheritance, by explaining the value of coding once and providing a means for devired objects to use this code.
Composition Versus Construction	Succeeds	Demonstrates	Inheritance	"Composition Versus Construction" demonstrates with coding how inheritance is used and its value to OOP.

Table 4-1: Sample Lesson Relationships

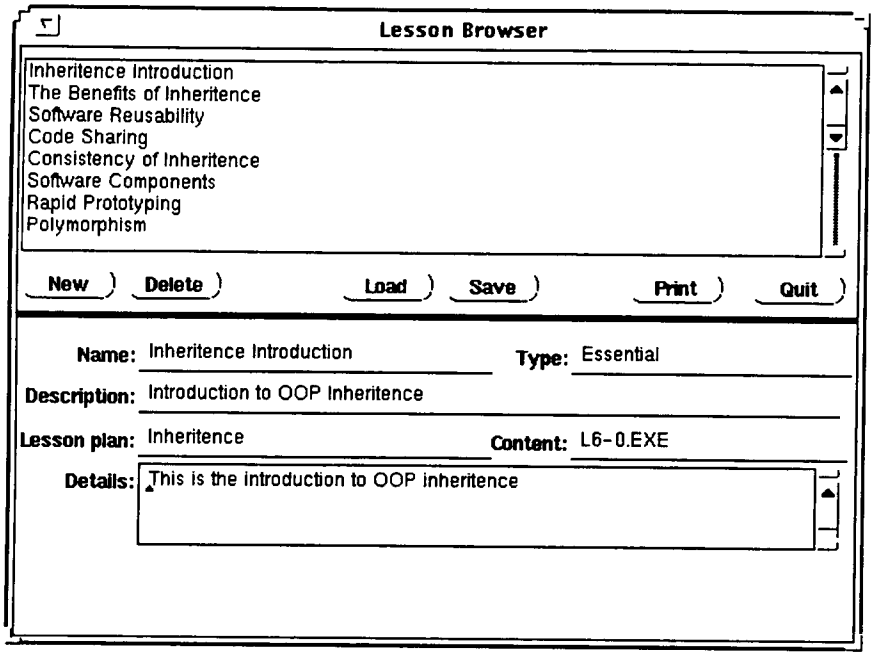


Figure 4-1: Example Lesson Browser Screen

The designer now executes the Lesson Plan Browser and selects "Add new plan..." from a pop-up menu and enters the information about the new Lesson Plan:

1. Name for this Lesson Plan: "Inheritance."
2. Short description of the plan: "OOP Inheritance."
3. Detailed description of the plan.

The designer assigns the desired lessons to this plan by selecting "Add new lesson..." from a pop-up menu and selects a lesson from a list of available lessons. This continues until all of the lessons for the plan are assigned.

The designer assigns relationships between the lessons in the plan by pressing the "Add" push button in the lesson relation data window. The DST asks for the two lessons

between which relationships are to be defined. Both lessons are selected from a list of available lessons. The DST then asks for the precedence of the first lesson with respect

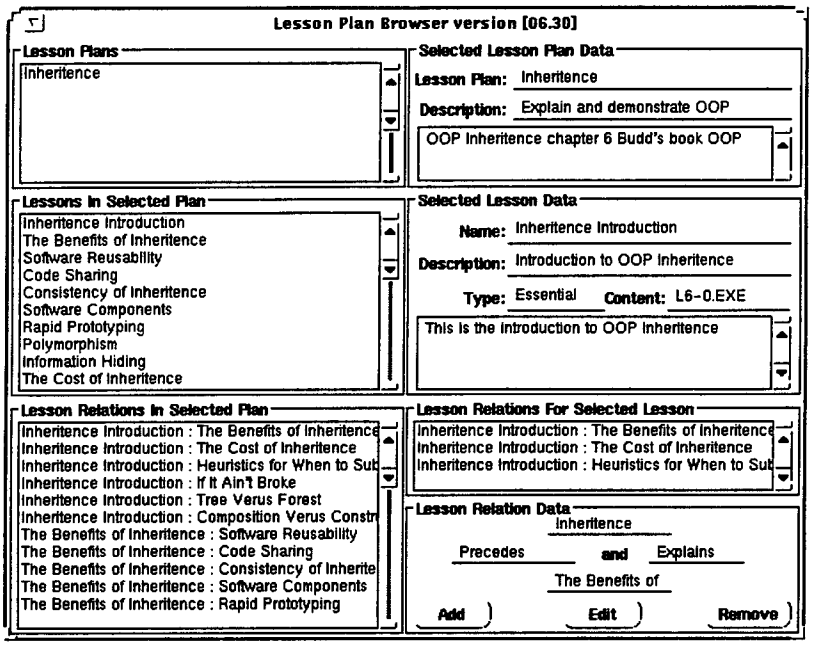


Figure 4-1: Example Lesson Plan

to the second lesson (only choices are "Precedes" or "Succeeds") and for the relation of the first lesson to the second lesson, selected from a list of relations. This procedure is repeated for all of the relationships between the lessons of the plan. At the end of the session the DST screen would look something like Figure 4-2.

After the relations are assigned, the designer selects "Compile relations..." from a pop-up menu; the DST searches through the lessons of the plan to build the Lesson Plan Tree Structure. If any inconsistencies are noted, the DST flags the designer and requires

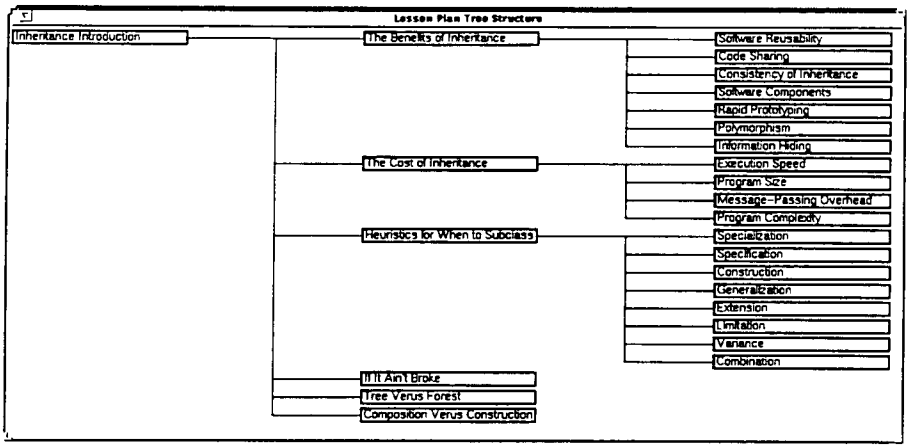


Figure 4-3: Example Lesson Plan Tree Structure Graph

the error to be corrected before it will continue with the compilation. This ensures that one error does not propagate additional errors throughout the tree structure.

Once the Lesson Plan Tree is compiled, the designer can request a graphical view of the tree (Figure 4-3). This aids in determining if the structure of the plan is correct and what to present to a student.

After the lesson plan is completed and successfully compiled, it can be assigned to students, using the Student Browser. The individual assigning lesson plans executes the Student Browser, and if the student is not already in the data base enters basic information about the student (name and social security number). The individual can then add or delete individual lesson plans for each student in the data base.

D. USING THE ITS

The above process would continue until the entire course is designed and the ITS is ready to be packaged. The complete ITS would include the individual Instructional

Units generated by an authoring tool, the knowledge-base rules/expert system, and the ITS moderator. The package would then be available on a designated system, and students would be assigned specific Lesson Plans. In this example, since each chapter is designed as a separate Lesson Plan, the instructor could assign Lesson Plans to students one at a time to ensure that each student completes the lessons in the correct sequence and does not attempt a more advanced lesson without having the basic information necessary. Since the ITS is designed to be used by sites without instructors, a specific individual is assigned to administer the training and maintain the system as required.

Once all of the components of the ITS (IUs, Knowledge Base, DST-generated lesson plans) are developed, a student can use the system to execute a course of instruction. The student accesses the ITS via the system moderator by logging onto the system (using their name and social security number). Once logged in, the student may select any of the assigned lesson plans to execute. Once a lesson plan is selected, the moderator presents the student with a list of the lessons in the plan and any lessons which the student has completed. The student selects a lesson to execute, and the moderator checks with the Student Information Module to see if the student has used this lesson plan and if the student is authorized to execute the requested lesson. If the request is authorized, the moderator loads the executable file and executes. During the execution of the IU, the moderator monitors the IU's execution for messages and interfaces with the knowledge base system to determine whether other information should be presented to the student. These procedures continue until the student has completed the IU or lesson

plan, or requests to end the tutoring session. The moderator records information about the student's system usage in the Student Information Module and continues with the course or terminates program execution.

V. CONCLUSIONS

A. SUMMARY

The future of education in DoD with decreasing personnel and funds will rely more and more on computer-aided instruction systems. The designers of these systems will require more advanced features (multimedia) and tools (CD ROM, etc.), and the ability to quickly design and maintain a tutoring system as requirements and the educational process evolve. Future tutoring systems will require intelligence to provide the support that human instructors currently are able to provide to the students. However, designing a specific Intelligent Tutoring System for a specific topic does not utilize the computer to its best advantage. The future of ITS must provide the ability to easily change or modify course contents and should use different components in parallel to provide the student with an educational tool which utilizes the power of modern computer systems. This will require the ITS to integrate lessons (IUs) generated with an advanced authoring system with a knowledge base expert system.

B. FUTURE RESEARCH

This thesis presented a bare-bones approach to the design of an Intelligent Tutoring System Design Support Tool. The proposed tool must be expanded and future research must determine the best system and test its interface with current authoring tools and expert systems.

1. Design of a support tool

The support tool should incorporate the design features stated in this thesis and provide an easy interface for the integration of the different components of an ITS. The proposed DST's capabilities need to be expanded to include the ability to link lessons together using a graphical interface. The best method would be to expand the functionality of the lesson plan tree structure graph. This expansion should include features which allow the designer to use point-and-click plus drag-and-drop techniques for linking and unlinking of lessons during the design process.

2. Design of knowledge base rules

Determining the rules for using the ITS is another major research project . The knowledge-based expert system tool should be simple enough for a nonprofessional designer to use and understand. The design tool must allow the designer to define specific rules based on the course objectives and the lesson outline. These rules will be used by the system moderator during the execution of the tutoring system.

3. Implementation of a complete ITS

The complete system should seamlessly integrate the three parts of the ITS: lessons (IUs), knowledge base, and student information module. This integration is accomplished with the system moderator. The moderator must be designed, coded and tested using the proposed features of the DST.

C. FINAL THOUGHTS

"As computer technology advances rapidly, computer systems become common tools for education. Although human expertise is better in teaching overall, computer tutors have the advantage of being available day and night. They never get tired and students find them less intimidating than [instructors]." [Ref. 8]

If downsizing continues and the use of advanced technology continues DoD must expand its use of computers in training and provide for a lack of human instructors with the use of ITS systems.

LIST OF REFERENCES

1. Bonar, J., Cunningham, R. and Schultz, J., "An Objected-Oriented Architecture for Intelligent Tutoring Systems, *OOPSLA '86 Proceedings*, pp. 269-76, SEPT. 1986.

2. Budd, T., An Introduction To Objected-Oriented Programming, Addison-Wesley, 1991.

3. Burns, H.L. and Capps, C.G., "Foundations of Intelligent Tutoring Systems", In M.C. Polsom & J.J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems*, pp. 1-19, LEA Publishers, New Jersey.

4. Goodman, B.A., "Multimedia Explanations for Intelligent Traingin Systems", *Conference on Intelligent Computer-Aided Training*, 1991.

5. Livergood, N.D., "Specification and Design Procedures, Functions, and Issues in Developing Intelligent Tutoring Systems", *J. ED. TECH.SYS.*, vol. 19(3), pp. 251-64, 1990-1991.

6. Russell, D.M., Moran, T.P. and Jordan, D.S., "The Instructional-Design Environment", In J. Psotka, L.D. Massey and S.A. Mutter (Eds), *Intelligent Tutoring Systems: Lessons Learned*, pp. 203-28, LEA Publishers, New Jersey.

7. Russell, D.M., "IDE: The Interpreter", In J. Psotka, L.D. Massey and S.A. Mutter (Eds), *Intelligent Tutoring Systems: Lessons Learned*, pp. 323-49, LEA Publishers, New Jersey.

8. Shim, Leemseop, "Student Modeling for an Intelligent Tutoring System: Based on the Analysis of Human Tutoring Sessions", Dissertation, April 1991, Illinois Institute of Technology.

9. Steier, D.M., et. al., "Combining Multiple Knowledge Sources in an Integrated Intelligent System", *IEEE Expert*, pp. 35-43, JUNE 1993.

10. Syang, A.A., "A Quantitative Student Model for Intelligent Tutoring Systems: Student Programming Ability", Dissertation, May 1992, University of Texas, Austin.

11. VaLehn, K., "Student Modeling", In M.C. Polsom & J.J. Richardson (Eds), *Foundations of Intelligent Tutoring Systems*, pp. 55-78, LEA Publishers, New Jersey.

12. Wong, S.T.C., "COSMO: A Communication Scheme for Cooperative Knowledge-based Systems", *IEEE Trans Systems*, vol. 23. pp. 809-24, MAY/JUNE 1993.

BIBLIOGRAPHY

Bonar, J., Cunningham, R. and Schultz, J., "An Object-Oriented Architecture for Intelligent Tutoring Systems, *OOPSLA '86 Proceedings*, pp. 269-76, SEPT. 1986.

Budd, T., An Introduction To Object-Oriented Programming, Addison-Wesley, 1991.

Burns, H.L. and Capps. C.G., "Foundations of Intelligent Tutoring Systems", In M.C. Polsom & J.J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems*, pp. 1-19, LEA Publishers, New Jersey.

Chen, J.W. and Chen M., "Toward the design of an Intelligent courseware production system using software engineering and industrial design principles", *J. ED. TECH. SYS.*, vol 19(1), pp. 41-52, 1990-91.

Goldberg, A., SmallTalk-80 The Interactive Programming Environment, Addison-Wesley, 1984.

Goldberg, A. and Robson, D., SmallTalk-80 The Language and its Implementation, Addison-Wesley, 1983.

Goodman, B.A., "Multimedia Explanations for Intelligent Training Systems", *Conference on Intelligent Computer-Aided Training*, 1991.

Livergood, N.D., "Specification and design procedures, functions, and issues in developing intelligent tutoring systems", *J. ED. TECH. SYS.*, vol 19(3), pp. 251-64, 1990-1991.

MacGreagor, R. and Burstein, M.H., "Using a description classification to enhance knowledge representation.", *IEEE Expert*, pp. 41-6, JUNE 1991.

Reid, J.C. and Mitchell, J. A., "The improvement of learning in computer assisted instruction.", *J. ED. TECH. SYS.*, vol 19(4), pp. 281-9, 1990-1991.

Russell, D.M., Moran, T.P. and Jordan, D.S., "The Instructional-Design Environment", In J. Psotka, L.D. Massey and S.A. Mutter (Eds.), pp. 203-28, *Intelligent Tutoring Systems: Lessons Learned*, LEA Publishers, New Jersey.

Russell, D.M., "IDE: The Interpreter", In J. Psotka, L.D. Massey and S.A. Mutter (Eds.), pp. 203-28, *Intelligent Tutoring Systems: Lessons Learned*, LEA Publishers, New Jersey.

Shim, Leemseop, "Student Modeling for an Intelligent Tutoring Stystem: Based on the Analysis of Human Tutoring Sessions", Dissertation, April 1991, Illinois Institute of Technology.

Steier, D.M. et. al., "Combining multiple knowledge sources in an integrated intelligent system.", *IEEE Expert*, pp. 35-43, JUNE 1993.

Swartout, W. and Moore, J., "Design for explainable expert systems.", *IEEE Expert*, pp. 58-64, JUNE 1991.

Syang, Anchir A., "A Quantitive Student Model for Intelligent Tutoring Systems: Student Programming Ability.", Dissertation, May 1992, Univerisity of Texas, Austin.

VaLehn, K., "Student Modeling", In M.C. Polsom & J.J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems*, pp. 55-78, LEA Publishers, New Jersey.

Wong, S. T. C., "COSMO: A Communication Scheme for Cooperative Knowledge-based Systems", *IEEE Trans Systems*, vol. 23, pp. 809-24, MAY/JUNE 1993.

INITIAL DISTRIBUTION LIST

	Number of Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 052 Naval Postgraduate School Monterey, California 93943-5002	2
3. Systems Management, Code 36 Naval Postgraduate School Monterey, California 93943-5002	1
4. Kishore Sengupta, Code SM/SE Department of Systems Management Naval Postgraduate School Monterey, California 93943-5002	5
5. B. Ramesh, Code SM/RA Department of Systems Management Naval Postgraduate School Monterey, California 93943-5002	1
6. LCDR Harry E. Landau Submarine Training Facility 544 White Road San Diego, California 92106-3550	2